Microcontroller Based Systems (CCE2014)
Study-Unit Handbook for Students

Department of Communications & Computer Engineering

2022–2023

Document r.8d04f1a1
Fri Feb 16 16:54:41 2024 +0100

Contents

1. Introduction

1.1. Assignment Overview

The assignment for the Microcontroller Based Systems study-unit is a group-based project where you will design and implement a complete system based on a current microcontroller. During the course of this assignment you will also gather experience in the use of a series of programming and documentation tools, such as the version control system SVN, debugging tools, the LaTeX type setting system and the DoxyGen source code documenting system.

The assignment requires considerable organisation on your part, as well as a lot of reading on your own and in your group. You will largely have to work out yourselves how to make best use of SVN and other tools, using the documentation that comes packaged with each tool. This involves a lot of reading and prototype programming to find out how the ARM MDK works in detail.

We will expect you to demonstrate that you have acquired the ability to understand and use tools that are new to you, and that you can apply knowledge from previous study-units. This is especially true for study-units such as:

- CCE1013 Computer Logic 1,
- CCE1014 Computer Logic 2,
- CCE2203 Signals and Systems,
- CPS1011 Programming Principles in C,
- CCE2111/CPS2004 Object Oriented Programming,
- ICS1018 Data Structures and Algorithms 1,
- CPS2002 Software Engineering.

However, feel free to take advantage of other study-units as well if it fits your project problem. Besides the 'technical' aspect of the project we also expect you to demonstrate ability in the project management and documentation side of the project. This means you also need to demonstrate that you are able to manage your group, your project, and most importantly yourself, as well as document your project and group process.

The assignment is a group project and all assessment will be based on your group as whole. Group marks will be moderated following intra-group peer assessment (see Section 4.4).

1.2. Task Outline

Your task is to develop a solution to a real-world problem using a microcontroller based system. The solution *must* fulfil a set of basic requirements, as detailed in Appendix B. Given the constraints imposed by the current worldwide situation, rather than leaving the assignment open-ended, a specific problem has been chosen for you to tackle, to ensure this is achievable with available or easily sourced hardware. You must not expect that reading all documentation listed in the reference section of this document will necessarily be sufficient to deliver a successful group project. References are just an indicator of where to start looking for further information.

There are two deliverables that are directly connected to the problem you're solving: a design brief (see Section 4.2) and a demonstration of your solution in action (Section 4.3). Besides coding, your group needs to manage the project well, according to established practice in software engineering. This means for example a structured coding style, code documentation using DoxyGen and keeping track of revisions of code using SVN (see Section C) as well as documenting group meetings in the form of a portfolio of meeting agendas and minutes (see Section 3.3). All these form part of your project documentation (another deliverable) that will be assessed at the end of the study-unit (Section 4.3), and must be kept up-to-date constantly.

2. Groups

2.1. Group Formation

Groups consist of 3 to 4 members and are *not* self-selected. Instead, they are allocated using a randomized process (for allocation see Section 2.3). Group members will meet each other during the first lecture (14 Feb 2024). You are expected to distribute the workload equitably between group members (and document this in the minutes of the group meetings).

2.2. Example Tasks within the Group

The project is such that no single person can handle all tasks on their own, due to the complexity of the problem and tools you need to get acquainted with. Hence, you *must* distribute different responsibilities and tasks within your group both during the design phase and the implementation phase. However, even though the work is divided, every group member should have a clear understanding the whole project, and be able to take over tasks from other group members as the need arises. A list of example tasks includes:

- reading of tool documentation (e.g. each member is responsible to explain one tool to the others)
- understand different parts of the ARM MDK
- implement different bits of the design
- integrating subsystems
- code documentation
- interfacing and related circuitry
- minute keeping
- project coordination

This list is neither prescriptive nor is it exhaustive, but provided simply to get an idea of what tasks there might be around. The distribution of tasks within the group is up to the group.

2.3. Group Allocation

Group 1: *i*) James John Gatt, *ii*) Lenise Silvio, *iii*) Ylenia Grima.

Group 2: *i*) Aleksandra Maria Krzemień, *ii*) Antonio Galdes, *iii*) Darren Lee Garrett, *iv*) Nathan Farrugia.

Group 3: *i*) Gabriel Carabott, *ii*) Isaac Grima, *iii*) Jacob Ellul, *iv*) Gioele Giunta.

Group 4: *i*) Alessandro Parrella, *ii*) Gabriel Vella, *iii*) Keith Farrugia, *iv*) Jonas Rousseau-Morvan.

Group 5: *i*) Graham Pellegrini, *ii*) Luca Vella, *iii*) Matthew Mifsud, *iv*) Julian Falzon.

Group 6: *i*) Carlston Azzopardi, *ii*) Luca Spiteri, *iii*) Rana Muhammad Kamran.

3. Delivery Pattern

The delivery of the study-unit consists of a series of lectures, lab sessions, group meetings and individual work, as shown below.

What?	*When?*	*Duration*	*Total Effort*
Lecture and Lab sessions	Weeks 1–10	3 hr weekly	30 hr
Group Meetings	Weeks 1–12	$1\frac{1}{2}$ hr weekly	18 hr
Individual Work	Weeks 1–12	$6\frac{1}{2}$ hr weekly	78 hr

Attendance is compulsory. Group members meet for the first time at the end of the first lecture to coordinate their first group meeting. Individual work (including group meetings) is expected to be about eight hours per week on average for each group member. This corresponds to an expected total average workload of 125 hours over the semester, and follows from the 5 ECTS gained in this study-unit. The necessary software is installed on computers in the Systems Interfacing lab, so they can be used for group and individual work.

3.1. Group Meetings

Groups are expected to have regular weekly meetings to discuss and take decisions related to the project, distribute and coordinate tasks, evaluate task outcomes, and resolve any technical or managerial issues. In order to ensure that meeting arrangements are acceptable to all members, meetings are to take place on campus[1]. Group members take turns chairing these meetings, which includes setting the agenda and timekeeping. Attendance at the meetings is compulsory and any apologies need to be communicated in advance of the meeting.

Groups are advised to find a time slot suitable to all group members for these meetings and stick to the chosen time slot through the semester to allow group members to plan ahead any extra-curricular activity. However do remember that you are registered as full-time students and therefore should be available at standard office hours throughout the week.

3.2. Agenda

In general group progress meetings will have the following agenda.

1. Date, Time, Place[2] of meeting.
2. Reading and approval of minutes from previous meeting[3].
3. Progress report – group members should detail progress in their responsibility area since the previous meeting[4], and targets may be set for the following week.
4. Any other set items the group has the wish to discuss.
5. Matters arising – group members then have the opportunity to discuss other matters (such as project-level difficulties).

Agendas should be produced and circulated to the group members and uploaded to SVN in advance of the meeting. Agenda items shall be numbered for easy reference in the minutes; a template is provided.

[1] Clearly, this only applies for in-person meetings, where these are necessary.
[2] Or video conferencing platform used.
[3] Except, of course, for the first meeting, where there are no previous minutes to approve
[4] *idem*

3.3. Minutes

Minutes of group meetings must include at least the following:

- Date, time, place and duration of meeting
- List of attendees and apologies (if any) for the absent members
- Approval of the minutes of the previous meeting
- Summary of discussion about and conclusion for items on the agenda.
- Actions for each member of the group with deadline. Actions needs to be numbered in the form $\mathrm{x.y}$ where x is the number of the week in which the meeting is held and y the running number of that action in that week. This make it easier to refer back to actions from previous meetings for follow-up.
- Follow-up of agreed actions in the last meeting.

Minutes must be uploaded to SVN[5] before your next meeting; they form part of the project documentation and will be assessed during the Final Audit (see Section 4). It is common practice to assign new actions to a member only when that member is present. This is so that the member with the new action can have their say. It is strongly recommended that you show your group's agendas and minutes to the lecturer, particularly in the first few weeks, for comments.

3.4. Group Communications

Important group communication shall take place using the members' University email accounts, and hence all members are expected to read and answer their emails at least once each working day. This is to prevent things like "Oh, I didn't know when the group meeting was because I never got your text" or "I did send you my apologies before the meeting, but my text message must have gotten lost." It also serves as evidence that communication has taken place in time, including all members, should a disagreement arise.

Files shall be shared exclusively using the SVN repository. This includes all source code, agendas, minutes, the design brief etc. It is not acceptable that one member uploads files to SVN on behalf of a different member, or files being shared by email or any other means, as this cannot be audited.

3.5. Design Phase

The first phase of your project, starting immediately, is the design phase. In this phase you have to get an overview of the ARM MDK components, decide how to implement a solution to the chosen problem, and verify that it is feasible to implement within the constraints of this class. In order to deliver a good project you must know all required tools well, so you can assess how hard or simple it will be to implement your project concept.

Taking into account the requirements detailed in Appendix B, you develop a system design to guide the implementation. The design brief due in 13 Mar 2024, 16:00 will presumably be at a high level, but shall be detailed enough to give your implementation a direction. At the very least, it needs to include enough detail to indicate which ARM MDK components need to be used (and therefore learnt in detail). As implementation proceeds, certain design decisions will have to be refined and/or revised as your experience with the ARM MDK grows. The continuously updated and revised design brief is a part of the project documentation and its versioning kept track of with SVN.

[5] and must not be changed after the date of approval!

3.6. Implementation Phase

Test code implementation should begin immediately, as soon as the groups are formed. To deliver a good design it is essential that as a group you have become acquainted with the tools you need to use *and* with the ARM MDK. With the ARM MDK it is particularly important that you understand how interfacing works, i.e. the code that has to be there for any embedded application to interface with external hardware.

Do adhere to good coding standards. DoxyGen is essential for other group members and yourself to understand your code. Both complete and meaningful DoxyGen comments and appropriate single-line comments to document small-scale implementation issues are a requirement for the project documentation. At the end of the implementation phase do not forget to test your system and check it against the requirements, Appendix B.

4. Assessment

Assessment consists of two components, based on the performance of your group as a whole. Note that assessment includes both your *product* (i.e. the final embedded system) as well as the *process* you followed to arrive at your product, and the process's documentation.

The Design Brief focuses on your product. The Final Audit focuses on the documentation of your design, development, and implementation processes, as well as your final product and video presentation. Finally, the Group Scaling Form (Appendix D.2) serves as a measure for each members' contribution to the group and hence measures each members' contribution to the group project as a whole.

4.1. Overview of Deadlines and Deliverables

All deliverables need to be committed to your group SVN repository by the deadlines below.

Deliverable	*Deadline*	*Assessed in:*
Design brief	13 Mar 2024, 16:00	Design Brief
Plagiarism form 1	13 Mar 2024, 16:00	Design Brief
Agenda	Group meeting	Final Audit
Minutes	Next group meeting	Final Audit
Design documentation	continuous until 24 May 2024, 16:00	Final Audit
Code and its documentation	continuous until 24 May 2024, 16:00	Final Audit
Demonstration Video	24 May 2024, 16:00	Final Audit
Plagiarism form 2	24 May 2024, 16:00	Final Audit
Group Scaling Form 1	20 Mar 2024, 16:00	Not assessed
Group Scaling Form 2	24 May 2024, 16:00	All Assessments

4.2. Initial Design Brief (25%)

Deadline: 13 Mar 2024, 16:00

Following your analysis of the ARM MDK, the chosen problem, and the requirements (Appendix B), you will produce a short report on the system design. This design brief is limited to a maximum of 4 pages. *No extra pages are allowed* for any additional material (this includes images, diagrams, references, appendices etc). Figures, and any text within them, must be large enough to be readable at print size – do not expect the reader to zoom in to read. A LaTeX template for this design brief is provided; *this template must not be changed in any way*, other than adding or removing sections. For details on the assessment criteria, please see the assessment form in Appendix D.1. Feedback will be uploaded to your SVN repository.

Note that the design brief needs to be authorised by all group members, i.e. all group members take responsibility for the design brief collectively. In practice this means for example all of them are equally responsible to avoid plagiarism. In view of this, together with the design brief itself, you will also need to submit a copy of the Declaration of Authenticity (Plagiarism Form) for assignments[6]. The form needs to be signed electronically[7] and submitted on SVN in the design brief folder, with filename `declaration1.pdf`. If any part of the submission, including the declaration of authenticity, is missing, the submission is considered late and will receive a zero grade.

Generally the design brief should demonstrate that the group has reflected about the chosen problem and how to implement a solution using a microcontroller system and the ARM MDK. It should reflect on how the group aims to implement the functional requirements and what the group expects to be the main challenges and difficulties during the project. The design brief shall make

[6] Available from the Faculty of ICT website at `https://www.um.edu.mt/ict/students/declarationofauthenticity`.
[7] Please refer to the instructions on the Group Scaling Form in D.2.

clear the feasibility of your project within the given time and resource constraints and your expertise. It should also include a verbal description of the chosen problem, with reference to how you plan to implement a microprocessor based solution, and also how you plan to transfer the requirements into software architecture. Your design brief should also contain comments on the following:

- How the solution to your chosen problem will be implemented using a microprocessor based system and the ARM MDK.
- How your design addresses the functional requirements in the Appendix.
- What additional input and output hardware are required, and how these will be sourced and connected.
- How to partition your software into individual components / classes.
- Interaction between different components of your hardware and software.
- Use of an appropriate development model: Rapid Software Development? Incremental Prototyping?
- What are the dependencies between different tasks?
- *A time plan.*

The Design Brief you submit will contain a System Design, which will be updated and kept as part of your project documentation towards a detailed final design. You will presumably need a series of test implementations and experiments with the microcontroller before you can arrive at a final design and implementation.

Note: The initial Design Brief is a separate deliverable; however its continuous updating is also part of the final project documentation which will be assessed as part of the Final Audit. After 13 Mar 2024, 16:00, the page limit is lifted, and you are expected to continue updating the design brief document in preparation for the Final Audit. For example, you would normally document how you tested your system, update the documentation to reflect the final design, and discuss any difficulties encountered, etc.

4.3. Final Audit (75%)

Deadline: 24 May 2024, 16:00

This is the final assessment of your project, and consists of a number of components:

1. A full audit, where *all* project code and documentation, together with the management of your project, will be assessed, as evidenced by work committed on SVN. Teamwork and management are assessed through the agendas, minutes, and your team's use of SVN (includes information from SVN logs and commits, as evidenced by `svn blame`). Documentation includes the continuously updated design documentation (which started as a design brief), as well as technical documentation using DoxyGen and other code comments. The page limit for the design documentation is lifted; however, there should not be any changes to the template. Only source code in `trunk` will be assessed (ie. at the time of the audit, your current communal version of the code has to reside in `trunk`).

   The project audit will be marked based on the quality and consistency of the project documentation, documentation of design and implementation decisions, and project progress with respect to its refined design and implementation. It is absolutely essential that *a fresh copy* of your project checked out from your repository at 24 May 2024, 16:00 compiles and can be loaded into the microcontroller without problems. Also DoxyGen must compile flawlessly.
2. Assessment of your final product against functional and technical requirements.
3. A short video presentation, containing a demonstration of your solution, and adhering to the following requirements:

- The video **must** be no longer than 5 min.
- This **must** be a single video file less than 200 MiB in size.
- Uploading large files is best done on campus from a wired connection. Failure to meet the deadline due to uploading problems is not considered a reasonable excuse.
- The video file **must** be named `groupXX.mp4` where `XX` is your two-digit group number.
- The video file **must** be playable with the VLC media player[8].
- The video file should be encoded as follows: a) container format has to be MP4, b) video should be encoded to H.264/MPEG-4 AVC[9] at Full HD resolution (1920×1080 progressive) with a frame rate of 25 fps (recommended maximum bit rate of 5 Mbps), and c) audio should be encoded to MPEG-4 AAC format (recommended maximum bit rate of 256 kbps).
- Any music or sound effects used in the video must be properly credited and permission to use these components must be obtained from the copyright holder. It is therefore recommended that any music or sound effects used should be either in the public domain or available on a compatible creative commons license.

Failure to meet any of the above requirements will disqualify the video in its entirety. The video presentation is to be submitted as follows.

a) Upload your video on Google Drive using your University account

b) Right-click on it and click 'Share...'

c) Input the email address of the study-unit lecturer (`johann.briffa@um.edu.mt`), and the addresses of the other members of your group. It is important that you do not amend default settings, i.e. that the lecturer 'can edit' and also that 'Notify people' is kept checked.

4. An assessment of your final solution, based on the evidence shown in the demonstration video.

For details on the assessment criteria, please see the assessment form in Appendix D.3. All the material submitted for the final assessment needs to be authorised by all group members, similarly to what was done for the design brief. In view of this, you will need to submit another copy of the Declaration of Authenticity (Plagiarism Form) for assignments[10]. The form needs to be signed electronically[11] and submitted on SVN in the design documentation folder, with filename `declaration2.pdf`. If any part of the submission, including the declaration of authenticity, is missing, the submission is considered late and will receive a zero grade.

4.4. Moderation of the Group Mark – Group Scaling

Assessment of all components of this study-unit is group-based and the group will be given a single mark for each piece of assessment. However this group mark is subject to individual scaling based on intra-group peer-assessment using the Group Scaling Form in Appendix D.2. This is where the group assess how well individual members contributed to the project.

The Group Scaling form will be used twice: the first is handed in by 20 Mar 2024, 16:00, following discussion during the day's lab session. This is a dry run that causes you to reflect on your group's functioning, identify problems and also to learn how scaling can effect each group member's individual mark. The second Group Scaling form will have to be handed in by 24 May 2024, 16:00, and will be used to scale the group marks of all components of assessment to yield individual marks.

[8] This can be downloaded from: `http://www.videolan.org/index.html`
[9] For a free encoder, go to: `http://www.videolan.org/developers/x264.html`
[10] Available from the Faculty of ICT website at `https://www.um.edu.mt/ict/students/declarationofauthenticity`.
[11] Please refer to the instructions on the Group Scaling Form in D.2.

A. Software Used

The following list contains recommended or required software for various aspects of your project. The software listed here is installed on the computers in the Systems Interfacing lab and/or the Networks Lab. You may also install all software on your own computer. All software used is available for free and runs under Windows and Linux[12]. Check the licences before installing. Be careful to follow the setup and configuration instructions accurately.

MDK-ARM Microcontroller Development Kit This provides the compilers, libraries, tools, and emulator.

Windows: `http://www.keil.com/arm/mdk.asp`

SVN Command-line and GUI interfaces

Linux: The command-line client (recommended) or kdesvn (GUI) are available in the standard repos.

Windows: The command-line client (recommended) can be installed in WSL[13], or TortoiseSVN (GUI) available from `http://tortoisesvn.tigris.org/`.

LaTeX For keeping agendas, minutes, and other documentation.

Linux: install package texlive-full, available in the standard repos.

Windows: install package texlive-full within WSL[13].

Editors Something to edit text files (such as LaTeX files)

Linux: It is recommended to use VS Code, available from `https://code.visualstudio.com/docs/setup/linux`. As an alternative, geany is available in the standard repos,

Windows: It is recommended to use VS Code, available from `https://code.visualstudio.com/docs/setup/windows`. Alternatively, use Notepad++, available from `http://notepad-plus.sourceforge.net/`.

Video and Audio Tools for recording and editing video and voice-overs

OBS Studio: Free and open source software for video recording and live streaming. Particularly useful for recording screen captures, which can be combined with voice over and live camera video. Available on Windows, Linux, and MacOS `https://obsproject.com/download`.

VLC media player: Free and open source multimedia player that is available on all major platforms `https://www.videolan.org/vlc/`.

Audacity: Free and open source software for audio recording and editing. Includes powerful audio processing tools, such as noise removal, etc. Available on Windows, Linux, and MacOS `https://www.audacityteam.org/download/`.

OpenShot: Free and open source video editor. Useful for cutting and editing video and audio assets and applying video effects. Fairly simple to use. Available on Windows, Linux, and MacOS `https://www.openshot.org/`.

Blender: Very powerful open source software for 2D and 3D content creation, which also includes a built-in video editor and compositor. Has a steep learning curve. Available on Windows, Linux, and MacOS `https://www.blender.org/`.

[12] Except for Keil, which is Windows only.
[13] Windows Subsystem for Linux, available at `https://docs.microsoft.com/en-us/windows/wsl/install`.

B. Requirements

The ARM MDK embedded system you are developing shall satisfy a number of functional and non-functional requirements. Some of them are formulated in a way that leaves room for your choice of implementation, and you have to elaborate yourselves how to best meet those requirements. Others are prescriptive and leave little or no space for interpretation. The non-functional requirements are split up again into technical and organisational requirements. In practice a hard line between functional and non-functional requirements can often not be drawn.

B.1. User Requirements

Develop a solution for a real-world application, built around an ARM Cortex-M4 microcontroller. The solution shall interact with the user, and must be easy and intuitive to use (no written documentation shall be needed for the user). The solution shall make good use of the system resources and must be aware of the memory and speed limitations that an embedded system imposes.

B.1.1. Chosen Problem

A specific problem has been chosen for you to tackle, to ensure this is achievable with available or easily sourced hardware. Specifically, your group is asked to implement a dual-tone multi-frequency signaling decoder with LCD output.

Dual-tone multi-frequency signaling (DTMF) is a telecommunication signaling system using the voice-frequency band over telephone lines between telephone equipment and other communications devices and switching centers. DTMF was first developed in the Bell System in the United States, and became known under the trademark Touch-Tone for use in push-button telephones supplied to telephone customers, starting in 1963. DTMF is standardized as ITU-T Recommendation Q.23 [1], and is also known in the UK as MF4. As the signals are audible tones in the voice frequency range, they can be transmitted through or recorded by any system that supports voice.

The DTMF system uses a set of eight audio frequencies transmitted in pairs to represent 16 signals, represented by the ten digits, the letters A to D, and the symbols # and *. In a typical implementation, the signals are chosen by the user through a keypad, arranged as three or four columns of four rows. Each row and column is identified by a different tone, so that each signal can be encoded by a combination of two distinct tones (one for the row and one for the column). As the signals originate from human input, the duration of each signal and the space between signals is undefined, and may vary within the same communication. For further details, it is recommended to start with Wikipedia[14].

A minimal implementation by your group will be able to take a line-level audio input of known fixed characteristics (including tone duration and spacing), decode this, and display the results on an LCD in real time. As time and ability allows, you may implement other features, such as (but not limited to):

- Automatic adaptation to different tone duration and spacing, with reliable decoding of sources where the timing is not precise, such as when the input is done by a human operator.
- Reliable decoding in the presence of noise (e.g. telephone line hiss or crackle).
- Automatic scrolling of text on LCD to handle messages of indefinite length.

You may also enhance your project by improving the interface, for example by building a suitable housing, using larger displays, integrating an audio monitor, etc. Note that the extent and quality of your implementation also impacts the grade you can obtain (c.f. Appendix D.3).

[14] https://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling

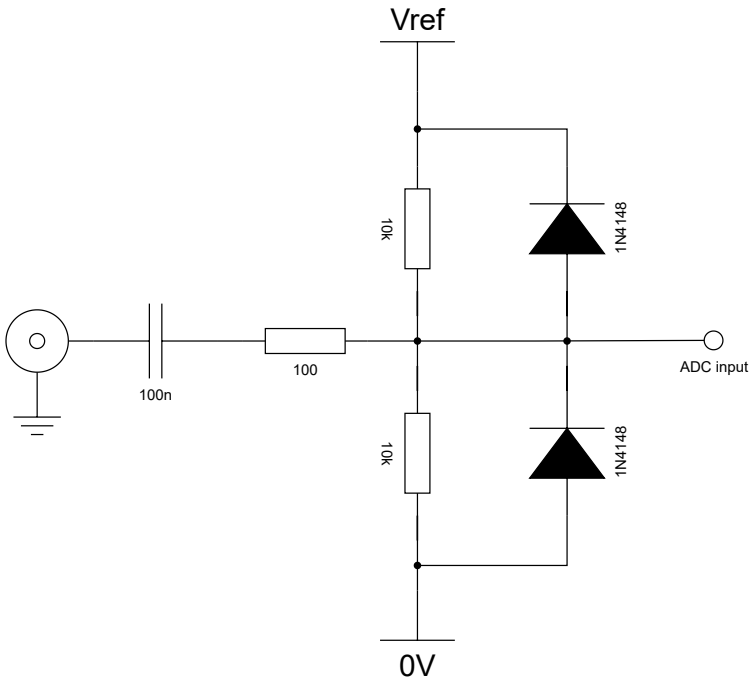

Figure 1: Example circuit to connect a line level output to the ADC input line.

B.1.2. Supporting Material

To facilitate testing your system, a set of sample keypad sequences and their corresponding DTMF-encoded audio files is provided. These are available for download from the VLE, as follows:

- Input keypad sequences
- Audio encodings:
  - Perfect timing – exact timing with no noise
  - Imperfect timing – small random variation in timing, with no noise
  - Noisy signal – small random variation in timing, with additive white noise

It is also recommended that you use a mobile app that can encode / generate arbitrary DTMF signals. An alternative is to use an offline encoder, such as `gen` from the `multimon` package for Linux.

The audio encodings can be played back on a laptop or mobile device, and connected to the microcontroller through the headphone or line output. The signal can be digitised on the microcontroller using the ADC peripheral. A suitable circuit to connect the headphone / line output to the ADC input line is shown in Figure 1. All necessary components are made available to implement this circuit.

B.2. System Requirements

B.2.1. Functional Requirements

(F.1) The solution must be real-time; that is, it must react to one or more inputs that can change at any time, in a way that feels immediate to the user.

(F.2) The system must make use of at least one digital input (e.g. a physical switch).

(F.3) The system must make use of at least one digital output (e.g. a LED).

(F.4) The system will connect with at least one external peripheral; connection may be through any of the available buses / interfaces (SPI, I2C, UART, USB, Ethernet). It is strongly recommended that this peripheral is a Hitachi HD44780-based 2-line text LCD display.

(F.5) On initial power up or reset, the system should enter a well-defined (and safe) state.

(F.6) The system shall have a persistent store for configuration settings, independent of the position of physical switches.

(F.7) The system shall have a useful and appropriate mechanism for indicating error conditions to the user.

(F.8) The system should be robust; that is, there should be no way for the system to enter an undefined or unstable state, even when given incorrect input.

B.2.2. Technical requirements

(T.1) Your solution will be based around the Embedded Artists LPC4088 experiment bundle.

(T.2) Your solution will be written entirely in C/C++ using the ARM MDK.

(T.3) Your solution will work on ARM MDK v5.16a onwards.

(T.4) No external libraries are allowed. However, the use of third-party code fragments is allowed where this: a) is clearly credited to the source, and b) is limited to a supportive role in the project (e.g. driver code).

(T.5) Use the limited system resources in a considerate and sustainable manner.

(T.6) The solution must be implemented as a multi-threaded system.

(T.7) The solution must make use of at least one interrupt handler.

B.2.3. Organisational Requirements

(O.1) Adhere to a clear and consistent coding style. That will make it easier for yourself, your group mates, and your examiners to understand your code.

(O.2) Use of the Keil µVision IDE is compulsory. The project (and its DoxyGen documentation) must compile without errors or warnings when checked out from SVN on the machines in the lab. Note that it is expected that the projects fit within the constraints of the free Lite / Evaluation edition, which you may install on your own machines. In any case, the licensed version is installed on the lab machines, and is available to all students.

(O.3) Write documentation of the code using DoxyGen.

(O.4) Use of the concurrent versioning system SVN is compulsory for code and all project documentation [2]. Commit log messages must be significant and useful.

(O.5) Minutes, Agendas and the design brief must be kept on the SVN server and written using LaTeX.

C. SVN Repository

C.1. Repository access

Each group has a repository at:

```
https://username@cce2014-ict.research.um.edu.mt/svn/CCE2014/2023-2024/Group_XX
```

where `XX` stands for your two-digit group number. This SVN server is accessible on- and off-campus.

C.2. Layout and file name conventions

The SVN repository layout has a specific folder structure that must be kept at all times.

code This folder holds the code of the project along with all documentation (DoxyGen) that is directly derived from the code.

- **trunk** The "latest communal version" of the project.– this is the only code version examiners will look at for marking purposes.
- **branches** Contains sub-folders for any branches from the main trunk, for example, for a developer working on a particular feature; branches are eventually merged into trunk and removed.
- **tags** Tagged specific versions of trunk

documentation This folder holds all documentation that is not directly derived from the code; it has the following sub-folders:

- **design** to hold the design brief, and possibly other technical documentation at the groups choice; the design brief must have the file name `design_brief.tex`
- **minutes** to hold the minutes; `minutes_weekYY.tex` where YY is the two digit number of the semester week (eg `minutes_week03.tex`) for the minutes of the supervised meeting in the third week of the semester. To avoid ambiguity, week numbers follow the dates for scheduled lectures, so that the deadlines are as follows (all deadlines at the time lecture starts):

Week	*Agenda Deadline*	*Minutes Deadline*
1	–	–
2	21 Feb 2024	28 Feb 2024
3	28 Feb 2024	6 Mar 2024
4	6 Mar 2024	13 Mar 2024
5	13 Mar 2024	20 Mar 2024
6	20 Mar 2024	10 Apr 2024
7	10 Apr 2024	17 Apr 2024
8	17 Apr 2024	24 Apr 2024
9	24 Apr 2024	8 May 2024
10	8 May 2024	–

  Only the first meeting in any given week will have its documentation assessed – these *must* be named as explained above. Any *additional* meetings in the same week should have a letter appended (eg `minutes_week03a.tex`). This list of dates is for the minimum number of meetings your group need to have. This documentation is checked for lateness. Any other meetings still need to be documented. However, the last meeting doesn't need to be minuted as you won't have the opportunity to approve these anyway.
- **agendas** to hold the agendas; the agendas have filenames as the minutes with 'minutes' replaced by 'agenda'.

feedback This directory shall not be changed by the group. It is used to post your project assessor's feedback after an assessment.

scaling This shall contain the electronic submission of the final group scaling form.

alien All files in the repository shall only hold work by the students. If you want to share material which is not yours using SVN it must be in the alien folder. All other files in the repository are assumed to be by the group (and your copyright).

test All files you wish not to be assessed in the Final Project Audit need to go here. It will not be looked at by the assessor. Note that this is not meant for 'work in progress', which should be assessed.

In addition the project holding your solution must be called "Group_XX" where XX is your two-digit group number.

D. Forms

Microcontroller Based Systems (CCE2014)
Design Brief

Prof. Johann A. Briffa

Group Number:

Description	*Grade*
Problem description and solution design concept: [10%]	
Engagement with functional requirements and realisation within ARM MDK: [30%]	
Initial planning for implementation: [30%]	
Organisation and management planning: [15%]	
Layout of report: [5%]	
Report template has been used correctly: [2%]	
Report does not exceed the maximum length: [2%]	
Figures in the report are legible, and any text readable, at print size: [2%]	
References are complete and correct: [2%]	
Report compiles correctly as given: [2%]	

Overall Grade:

Briefly justify the overall grade, and give additional feedback.

Instructions

The group examiner should complete this assessment and submit it on the following folder on SVN:

`https://username@cce2014-ict.research.um.edu.mt/svn/CCE2014/2023-2024/assessment/designbrief/`

The filename should be `designbrief-XX.pdf`, where `XX` is the two-digit group number. *Please follow the naming*

Academic Year 2023–2024 Form r.1d379d99, Fri Mar 15 14:51:56 2024 +0100

convention strictly as a script will be used to collect grades. All grades are provisional until Board of Examiners approval.

Marking Criteria

Grade *Description*

Problem description and solution design concept: [10%]

0%–44%: An insignificant or limited attempt at presenting the design concept, or the presented concept shows a misunderstanding of the task. The design is either not sufficiently developed or difficult to understand and misrepresents the chosen problem.
45%–49%: A design concept has been presented, but it lacks thoroughness and coherence. Its description has significant omissions and/or inconsistencies.
50%–54%: The design concept is thought through but has significant shortcomings. Its is described adequately but with some omissions and/or inconsistencies.
55%–59%: The design concept is thought through but has some shortcomings. Its is described adequately.
60%–69%: The design concept is well thought through and addresses at least the minimum required specification. Its is mainly described coherently and concisely with only minor shortcomings.
70%–74%: The design concept is well thought through and addresses the minimum required specification and some additional features. Its is described coherently and concisely.
75%–79%: The design concept is very well thought through and comprehensively addresses the minimum required specification and additional features. It is described coherently and concisely.
80%–89%: The design concept is very well thought through and comprehensively addresses the minimum required specification and additional features. It is described coherently and concisely, showing critical understanding.
90%–100%: The design concept is exceptionally well presented and comprehensively addresses the minimum required specification and additional features. It is described coherently and concisely, showing critical understanding.

Engagement with functional requirements and realisation within ARM MDK: [30%]

0%–44%: Insignificant or inappropriate attempt to address functional requirements, or some functional requirements are addressed but with limited scope and with significant shortcomings and oversights. No significant awareness of connection to the ARM MDK.
45%–49%: Some functional requirements are addressed but with limited scope and with some significant shortcomings and oversights. The connection between the functional requirements and the ARM MDK is poor.
50%–54%: Functional requirements are addressed adequately and a connection made to the ARM MDK but with significant shortcomings and oversights.
55%–59%: Functional requirements are addressed adequately and a connection made to the ARM MDK but with some shortcomings and oversights.
60%–69%: Functional requirements are addressed adequately and a detailed connection made to the ARM MDK with some limited oversights and shortcomings.
70%–74%: Functional requirements are addressed well and a detailed connection made to the ARM MDK with only minor oversights and shortcomings.
75%–79%: Functional requirements are addressed well, concisely and comprehensively with an informed and detailed connection made to the ARM MDK with only minor oversights and shortcomings.
80%–89%: Functional requirements are addressed very well, concisely and comprehensively with a thorough and detailed connection made to the ARM MDK.
90%–100%: Functional requirements are addressed very well, concisely and comprehensively with a thorough and detailed connection made to the ARM MDK.

Initial planning for implementation: [30%]

(Continued on next page)

(Continued from previous page)

Grade	*Description*
0%–44%:	An insignificant or inadequate initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation with severe omissions and flaws and limited awareness of the implementation.
45%–49%:	A limited initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation with a few severe oversights and shortcomings.
50%–54%:	A barely adequate initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation with some significant oversights and shortcomings.
55%–59%:	An adequate initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation with some oversights and shortcomings.
60%–69%:	A good professional initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation with only some oversights and shortcomings.
70%–74%:	A good professional initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation with only minor oversights and shortcomings.
75%–79%:	A thorough professional initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation with only minor oversights and shortcomings.
80%–89%:	An excellent and very thorough professional initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation.
90%–100%:	An exceptional and very thorough professional initial implementation plan regarding user interface, hardware design and object-oriented structure of the implementation.

Organisation and management planning: [15%]

0%–44%:	An insufficient approach to task identification and time planning with minimal understanding and presentation of task interdependencies and with severe flaws or omissions.
45%–49%:	A limited approach to task identification and time planning with restricted understanding and presentation of task interdependencies and with severe flaws.
50%–54%:	An adequate approach to task identification and time planning with some understanding and presentation of task interdependencies, but with significant flaws.
55%–59%:	An adequate approach to task identification and time planning with some understanding and presentation of task interdependencies, but with minor flaws.
60%–69%:	A good approach to task identification and time planning with appropriate presentation of task interdependencies and with only some flaws.
70%–74%:	A good approach to task identification and time planning with appropriate presentation of task interdependencies and with only minor flaws.
75%–79%:	A thorough approach of near professional standard to task identification and time planning with a professional understanding and presentation of task interdependencies and only a few minor flaws.
80%–89%:	An excellent and very thorough approach of professional standard to task identification and time planning with a professional understanding and presentation of task interdependencies.
90%–100%:	An exceptional and very thorough approach of professional standard to task identification and time planning with a professional understanding and presentation of task interdependencies.

Layout of report: [5%]

0%–44%:	An unorganised layout and organisation of material with significant design and readability errors.
45%–49%:	A layout and organisation of material with severe shortcomings.
50%–54%:	An adequate layout and organisation of material with significant shortcomings.
55%–59%:	An adequate layout and organisation of material with some shortcomings.
60%–69%:	A good layout and organisation of material with some shortcomings.
70%–74%:	A good layout and organisation of material with only minor shortcomings.

(Continued on next page)

(Continued from previous page)

Grade	*Description*
75%–79%:	A very good layout and organisation of material with a few minor shortcomings.
80%–89%:	A professional layout and organisation of material.
90%–100%:	A professional layout and organisation of material.

Microcontroller Based Systems (CCE2014)
Group Scaling Form

Prof. Johann A. Briffa

Group Number:

Name	*ID*	*Contrib.*	*Date*	*Signature*

Instructions

Completing and Submitting this Form

Please complete this form electronically with Adobe Acrobat Reader DC[1]. Once all other details are filled in, signatures need to be added by each group member by clicking 'fill and sign' and indicating that you're the one signing. No amendments are allowed on the form. The completed form is to be submitted on the following folder on SVN:

`https://username@cce2014-ict.research.um.edu.mt/svn/CCE2014/2023-2024/Group_XX/documentation/scaling/`

where XX is your two-digit group number, and the filename should be:

- `group-scaling-final.pdf` for the final submission (deadline for submission: 24 May 2024, 16:00), and
- `group-scaling-dryrun.pdf` for the dry-run, (deadline for submission: 20 Mar 2024, 16:00).

Please follow the naming convention strictly as a script will be used to collect grades. Also note that the submitted form should be the one completed electronically with Acrobat. No other method or software is allowed.

Dry-Run

A dry-run of this process is held during the lab session on the day of the deadline. In preparation for this you should complete and submit a copy of the form and get it with you for discussion.

[1] This is available for free from `https://get.adobe.com/reader/` for Mac and Windows users. Your signature needs to be entered once for every installation of Reader, using a graphic tablet or by scanning it in.

How This Works

The contribution value listed for each group member indicates the individual contribution to the group work; specifically:

- A score of 0 indicates that the group member contributed to the group as expected, doing no more or no less than was agreed.
- A value less than 0 indicates that the group member did less work than expected, with a more negative number indicating less effort.
- A value greater than 0 indicates that the group member did more work than expected, with a more positive number indicating more effort.

Note that the sum of the individual contributions for the group members must be 0. This means that if one or more people have negative scores, then there must be one or more people having positive scores to balance this out and vice-versa. Each member's score will be used to scale the overall mark the group achieved for that member by fractions of ten.[2] For example, if the overall group mark is 60% and

- a member gets a score of -2, then that member will get a final mark of

$$60\% \times \frac{10-2}{10} = 48\%$$

- a member gets a score of $+2$, then that member will get a final mark of

$$60\% \times \frac{10+2}{10} = 72\%$$

All grades are provisional until Board of Examiners approval.

[2] This is subject to the obvious constraint that scaled marks are clipped at 100% and 0%, i.e. nobody can get a mark less than 0% or greater than 100%.

Microcontroller Based Systems (CCE2014)
Final Audit

Prof. Johann A. Briffa

Group Number: ☐

Description	*Grade*
Team Management – agendas: [6%]	☐
Agendas source files follow naming conventions meticulously: [1%]	
Team Management – minutes: [6%]	☐
Minutes source files follow naming conventions meticulously: [1%]	
Use of SVN:	
Regular and consistent use of SVN, each commit representing a small conceptual update: [2%]	
Individual group members contributed equitably, as evidenced by SVN commits: [2%]	
SVN commit messages are descriptive and to the point: [2%]	
Design documentation: [10%]	☐
Design brief sources follow naming conventions meticulously: [1%]	
Figures in the report are legible, and any text readable, at print size: [1%]	
References are complete and correct: [1%]	
Report compiles correctly as given: [1%]	
Code comments and technical documentation: [10%]	☐
Structure of source code: [5%]	☐
Code folder structure follows naming conventions meticulously: [1%]	
Project trunk:	
Project checked out from trunk of the group's SVN repository compiles without errors: [3%]	
Project also compiles without warnings: [1%]	
Functional and Technical Requirements:	
Solution is real-time (reacts immediately to asynchronous user inputs): [1%]	
System has at least one digital input (e.g. a switch): [1%]	
System has at least one digital output (e.g. an LED): [1%]	
System connects to at least one peripheral (e.g. text LCD display): [1%]	
On initial power up, system enters a well-defined stable state: [1%]	
System configuration is persistent between reboots: [1%]	
Error conditions are indicated clearly to the user: [1%]	

(Continued on next page)

Academic Year 2023–2024 Form r.1d379d99, Fri Mar 15 14:51:56 2024 +0100

(Continued from previous page)

Description	*Grade*
System has been tested for robustness, even with incorrect input: [1%]	
System resources used in a considerate and sustainable manner: [1%]	
Solution implemented as a multi-threaded system: [1%]	
Solution makes use of at least one interrupt handler: [1%]	
Video Presentation: [8%]	
Video adheres to required specifications (filename, length, size, codec, bitrate, and container format): [2%]	
Final product:	
Implementation of standard features (as required for a minimal working system): [8%]	
Extra features (rate here any other elements not included above, or not strictly required): [8%]	
Presentation (rate the aesthetics of the solution for appropriateness and finish): [3%]	
Usability of controls: [3%]	
Intuitiveness (i.e. how much of the system could you use before needing help?): [3%]	

Overall Grade:

Briefly justify the overall grade, and give additional feedback.

Instructions

The group examiner should complete this assessment, based on evidence committed on SVN and on the video presentation, as appropriate. Criteria marked on a Likert scale should be interpreted as follows:

Excellent Surpasses expectations; consistently exceeds requirements

Good Meets all requirements, may occasionally exceed expected level

Fair Generally meets expected level; some deficiencies may exist but none of major concern

Poor Generally fails to meet expected level; significant deficiencies

Very Poor Significantly below expectations; many deficiencies

Missing Applies only when the given aspect has not been addressed

When complete, this is to be submitted on the following folder on SVN:

`https://username@cce2014-ict.research.um.edu.mt/svn/CCE2014/2023-2024/assessment/audit-final/`

The filename should be `audit-final-XX.pdf`, where `XX` is the two-digit group number. *Please follow the naming convention strictly as a script will be used to collect grades.* All grades are provisional until Board of Examiners approval.

Academic Year 2023–2024 Form r.1d379d99, Fri Mar 15 14:51:56 2024 +0100

Marking Criteria

Grade *Description*

Team Management – agendas: [6%]

0%–44%: Insufficient planning for meetings. Agendas often missing or contain only very vague or generic points. There is usually little to differentiate one meeting from another.

45%–49%: Limited planning for meetings. Some agendas missing; the others mostly consist of standard items for discussion only.

50%–54%: Basic planning only. Agendas submitted for every meeting, though some may be submitted late. Items for discussion are mostly standard items; some may be unclear.

55%–59%: Basic planning only. Agendas submitted for every meeting, though some may be submitted late. Items for discussion are mostly standard items, but are generally clear.

60%–69%: Planning was adequate. Agendas submitted for every meeting, and are always on time. Items for discussion are mostly standard items, but are generally clear.

70%–74%: Planning was at a near professional level and reflects good organisational skills. Agendas submitted for every meeting, and are always on time. Items for discussion are mostly standard items, but are generally clear.

75%–79%: Planning was at a professional level and reflects very good organisational skills. Agendas submitted for every meeting, and are always on time. Items for discussion are generally clear.

80%–89%: Planning was at a professional level and reflects excellent organisational skills. Agendas submitted for every meeting, and are always on time. Items for discussion are always clear and substantive.

90%–100%: Planning was at a professional level and reflects exceptional organisational skills. Agendas submitted for every meeting, and are always on time. Items for discussion are always clear and substantive.

Team Management – minutes: [6%]

0%–44%: Insufficient documentation of meetings. Minutes often missing or contain only very vague or generic discussion points. There is usually little to differentiate one meeting from another.

45%–49%: Limited documentation of meetings. Some minutes missing; the others mostly consist of standard discussion points only.

50%–54%: Basic documentation of meetings only. Minutes submitted for every meeting, though some may be submitted late. Descriptions of discussion may be unclear.

55%–59%: Basic documentation of meetings only. Minutes submitted for every meeting, and are always on time. Descriptions of discussion may occasionally be unclear.

60%–69%: Documentation of meetings was adequate. Minutes submitted for every meeting, and are always on time. Descriptions of discussion are generally clear. Action points, listed separately, may be unclear or occasionally missing.

70%–74%: Documentation of meetings was at a near professional level and reflects good organisational skills. Minutes submitted for every meeting, and are always on time. Descriptions of discussion are generally clear. Action points, listed separately, may be unclear or occasionally missing.

75%–79%: Documentation of meetings was at a professional level and reflects very good organisational skills. Minutes submitted for every meeting, and are always on time. Descriptions of discussion are generally clear. Action points, listed separately, are generally clear.

80%–89%: Documentation of meetings was at a professional level and reflects excellent organisational skills. Minutes submitted for every meeting, and are always on time. Descriptions of discussion are always clear and substantive. Action points, listed separately, are always clear.

90%–100%: Documentation of meetings was at a professional level and reflects exceptional organisational skills. Minutes submitted for every meeting, and are always on time. Descriptions of discussion are always clear and substantive. Action points, listed separately, are always clear.

Design documentation: [10%]

(Continued on next page)

(Continued from previous page)

Grade	*Description*
0%–44%:	The limited documentation provided is incoherent and/or misrepresents substantial parts of the project.
45%–49%:	The documentation is maintained at a limited level and reflects a limited understanding of the project with inconsistencies and/or omissions.
50%–54%:	The documentation is maintained and reflects a basic understanding of the project with some significant inconsistencies and/or omissions.
55%–59%:	The documentation is maintained and reflects a basic understanding of the project with some inconsistencies and/or omissions.
60%–69%:	The documentation is maintained at a near professional level and reflects an adequate understanding of the project with some inconsistencies and/or omissions.
70%–74%:	The documentation is maintained at a near professional level and reflects a good understanding of the project with only some minor inconsistencies and/or omissions.
75%–79%:	The complete documentation is maintained at a professional level and reflects a substantive understanding of the project.
80%–89%:	The complete documentation is maintained at a professional level and reflects an excellent understanding of the project.
90%–100%:	The complete documentation is maintained at a professional level and reflects an exceptional understanding of the project.

Code comments and technical documentation: [10%]

0%–44%:	The code is commented in an incoherent or misleading way. Many comments are lacking or do not contain relevant information.
45%–49%:	The code is commented at a limited level with inconsistencies and/or omissions. Comments are not concise and complete.
50%–54%:	The code is commented, both using Doxygen and normal comments, however with some significant inconsistencies and/or omissions. Comments can be improved with respect to relevance, conciseness and completeness. They may have some shortcomings.
55%–59%:	The code is commented, both using Doxygen and normal comments, however with some inconsistencies and/or omissions. Comments can be improved with respect to relevance, conciseness and completeness. They may have some shortcomings.
60%–69%:	The code is commented adequately, both using Doxygen and normal comments. Comments are mostly relevant, concise and complete, and may have minor shortcomings.
70%–74%:	The code is commented at a near professional standard, both using Doxygen and normal comments. Comments are mostly relevant, concise and complete, and may have minor shortcomings.
75%–79%:	The code is commented at good professional standard, both using Doxygen and normal comments. Comments are relevant, concise and complete. Doxygen comments are syntactically correct.
80%–89%:	The code is commented at an excellent professional standard, both using Doxygen and normal comments. Comments are relevant, concise and complete. Doxygen comments are syntactically correct.
90%–100%:	The code is commented at an exceptional professional standard, both using Doxygen and normal comments. Comments are relevant, concise and complete. Doxygen comments are syntactically correct.

Structure of source code: [5%]

0%–44%:	A limited amount of code is provided and the code mainly fails to follow professional object-oriented practice. The lack of clear structure severely impacts readability. Alternatively, too little code has been provided to assess adherence to object-oriented practice, structure and readability.
45%–49%:	A limited amount of code is provided and the code follows professional object-oriented practice with considerable shortcomings. The code is only partly structured, affecting readability.
50%–54%:	A fair amount of code is provided and the code follows professional object-oriented practice with some significant shortcomings. The code is only partly structured, but still readable.

(Continued on next page)

(Continued from previous page)

Grade	*Description*
55%–59%:	A fair amount of code is provided and the code follows professional object-oriented practice with some shortcomings. The code is only partly structured, but still readable.
60%–69%:	An appropriate amount of code is provided and the code follows professional object-oriented practice with a few shortcomings. The code is mostly well structured and readable.
70%–74%:	An appropriate amount of code is provided and the code follows professional object-oriented practice with a few minor shortcomings. The code is mostly well structured and readable.
75%–79%:	An appropriate amount of code is provided and the code follows professional object-oriented practice to an advanced degree. The code is very well structured and readable.
80%–89%:	An appropriate amount of code is provided and the code follows professional object-oriented practice to an excellent degree. The code is excellently structured and readable.
90%–100%:	An appropriate amount of code is provided and the code follows professional object-oriented practice to an exceptional degree. The code is exceptionally structured and readable.

Video Presentation: [8%]

0%–44%:	The video submitted (if any) is incoherent, does not cover, and/or misrepresents substantial parts of the project.
45%–49%:	The video submitted is limited and barely covers basic aspects of the project.
50%–54%:	The video submitted reflects a satisfactory understanding of the project with some shortcomings (omissions and/or inconsistencies).
55%–59%:	The video submitted reflects a satisfactory understanding of the project. Presentation quality is adequate.
60%–69%:	The video submitted reflects a sound understanding of the project. Presentation quality is average.
70%–74%:	The video submitted reflects a substantial understanding of the project. Presentation quality is better than average.
75%–79%:	The video submitted reflects a comprehensive understanding of the project. Presentation quality is good, but unremarkable.
80%–89%:	The video submitted reflects a comprehensive and critical understanding of the project. Presentation quality is excellent, and comes across as polished.
90%–100%:	The video submitted reflects an exceptional understanding of the project. Presentation quality is exceptional.

References

[1] "Technical features of push-button telephone sets," International Telecommunication Union, Geneva, CH, Recommendation ITU-R Q.23 (11/1988), 1993. [Online]. Available: http://www.itu.int/rec/T-REC-Q.23/en

[2] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato, "Version control with subversion," 2011, previous editions have been published with O'Reilly. [Online]. Available: http://svnbook.red-bean.com/en/1.7/svn-book.pdf